Introduction to the Cray X86 Compiler

Nathan Wichmann

wichmann@cray.com

CRAY
THE SUPERCOMPUTER COMPANY

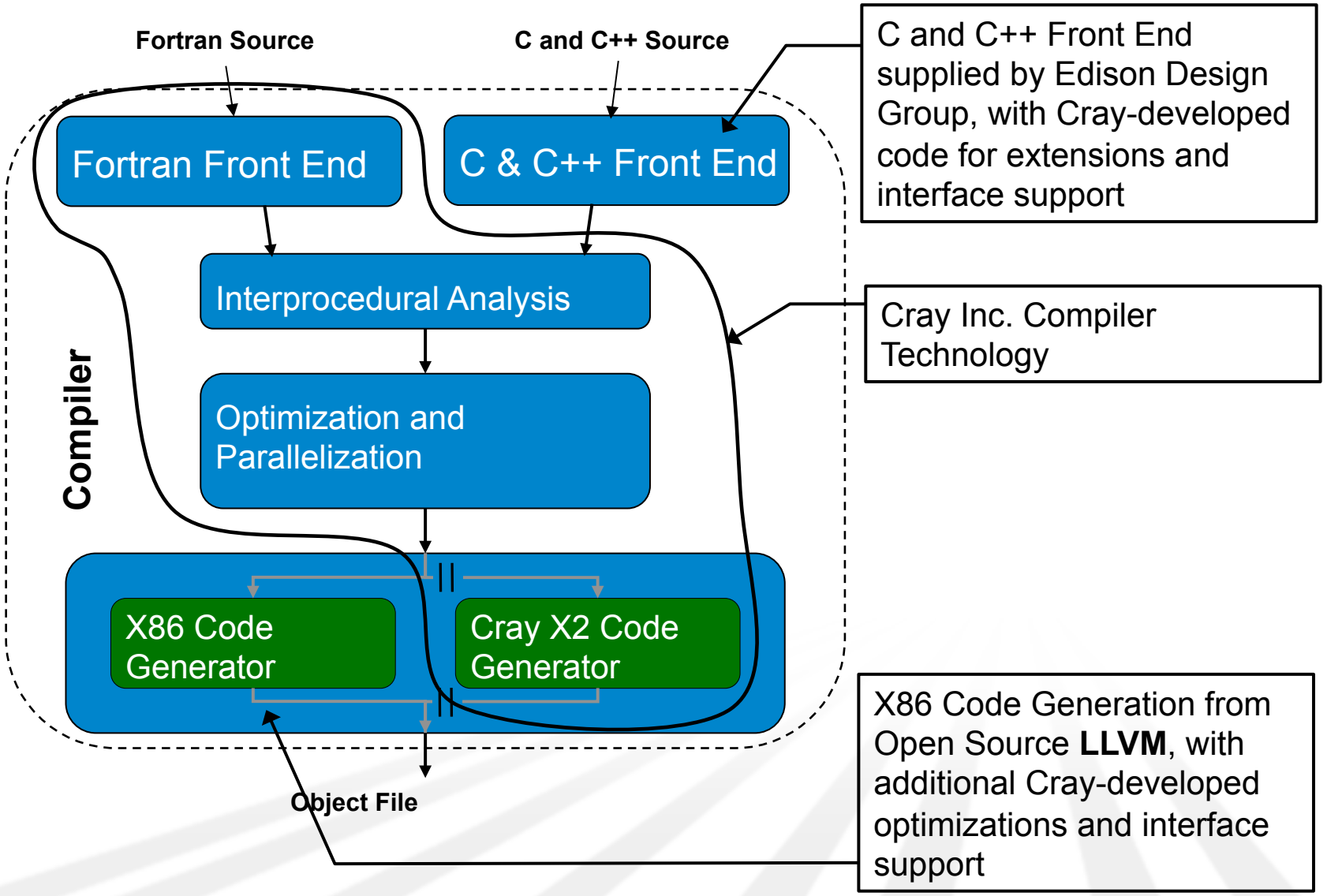
Cray Opteron Compiler: Brief History of Time

- Cray has a long tradition of high performance compilers
  - Vectorization
  - Parallelization
  - Code transformation
  - More…
- Began internal investigation leveraging an open source compiler called LLVM
- Initial results and progress better than expected
- Decided to move forward with Cray X86 compiler
- First release December 2008

Why a Cray X86 Compiler?

- Standard conforming languages and programming models
  - Fortran 2003
  - UPC & CoArray Fortran
    - **Fully optimized** and integrated into the compiler
    - No preprocessor involved
    - Target the network appropriately:
      - GASNet with Portals
      - DMAPP with Gemini & Aries
- Ability and motivation to provide high-quality support for custom Cray network hardware
- Cray technology focused on scientific applications
  - Takes advantage of Cray's extensive knowledge of **automatic vectorization**
  - Takes advantage of Cray's extensive knowledge of **automatic shared memory parallelization**
  - **Supplements**, rather than replaces, the available compiler choices

Technology Sources



Cray Opteron Compiler: How to use it

- Make sure it is available
  - module avail PrgEnv-cray
- To access the Cray compiler
  - module load PrgEnv-cray
- To target the Barcelona chip
  - module load xtpe-quadcore
- Once you have loaded the module “cc” and “ftn” are the Cray compilers
  - Recommend just using default options
  - Use –rm (fortran) and –hlist=m (C) to find out what happened

Example: ftn –rm –c file.f90

Cray Opteron Compiler: Current Capabilities

- Excellent Vectorization
  - Vectorize more loops than other compilers
- OpenMP
  - 2.0 standard
  - Nesting
- PGAS: Functional UPC and CAF available today.
- Excellent Cache optimizations
  - Automatic Blocking
  - Automatic Management of what stays in cache
- Prefetching, Interchange, Fusion, and much more…

Cray Opteron Compiler: Directives

- Cray compiler supports a full and growing set of directives and pragmas

!dir$ concurrent
!dir$ ivdep
!dir$ interchange
!dir$ unroll
!dir$ loop_info [max_trips] [cache_na] ... Many more
!dir$ blockable

man directives

man loop_info

Cray Opteron Compiler: Current Strengths

- Loop Based Optimizations
  - Vectorization
  - Interchange
  - Pattern Matching
  - Cache blocking/ non-temporal / prefetching
- Fortran Standard
- PGAS (UPC and Co-Array Fortran)
- Optimization Feedback: Loopmark
- Focus

Loopmark: Compiler Feedback

- Compiler can generate an filename.lst file.
  - Contains annotated listing of your source code with letter indicating important optimizations

```
%%%   L o o p m a r k   L e g e n d   %%%
   Primary Loop Type      Modifiers
   ------- ---- ----      ---------
                          a - vector atomic memory operation
   A  - Pattern matched     b - blocked
   C  - Collapsed           f - fused
   D  - Deleted             i - interchanged
   E  - Cloned              m - streamed but not partitioned
   I  - Inlined             p - conditional, partial and/or computed
   M  - Multithreaded       r - unrolled
   P  - Parallel/Tasked     s - shortloop
   V  - Vectorized          t - array syntax temp used
   W  - Unwound             w - unwound
```

Example: Cray loopmark messages for Resid

- ftn –rm … or cc –hlist=m …

```
29.  b-------<      do i3=2,n3-1
30.  b b-----<        do i2=2,n2-1
31.  b b Vr--<           do i1=1,n1
32.  b b Vr               u1(i1) = u(i1,i2-1,i3) + u(i1,i2+1,i3)
33.  b b Vr      >              + u(i1,i2,i3-1) + u(i1,i2,i3+1)
34.  b b Vr               u2(i1) = u(i1,i2-1,i3-1) + u(i1,i2+1,i3-1)
35.  b b Vr      >              + u(i1,i2-1,i3+1) + u(i1,i2+1,i3+1)
36.  b b Vr-->           enddo
37.  b b Vr--<           do i1=2,n1-1
38.  b b Vr               r(i1,i2,i3) = v(i1,i2,i3)
39.  b b Vr      >                - a(0) * u(i1,i2,i3)
40.  b b Vr      >                - a(2) * ( u2(i1) + u1(i1-1) + u1(i1+1) )
41.  b b Vr      >                - a(3) * ( u2(i1-1) + u2(i1+1) )
42.  b b Vr-->           enddo
43.  b b----->        enddo
44.  b------->      enddo
```

Example: Cray loopmark messages for Resid (cont)

ftn-6289 ftn: VECTOR File = resid.f, Line = 29

*A loop starting at **line 29 was not vectorized** because a recurrence was found on "U1" between lines 32 and 38.*

ftn-6049 ftn: SCALAR File = resid.f, Line = 29

*A loop starting **at line 29 was blocked with block size 4.***

ftn-6289 ftn: VECTOR File = resid.f, Line = 30

A loop starting at line 30 was not vectorized because a recurrence was found on "U1" between lines 32 and 38.

ftn-6049 ftn: SCALAR File = resid.f, Line = 30

A loop starting at line 30 was blocked with block size 4.

ftn-6005 ftn: SCALAR File = resid.f, Line = 31

*A loop starting at **line 31 was unrolled 4 times.***

ftn-6204 ftn: VECTOR File = resid.f, Line = 31

*A loop starting at **line 31 was vectorized.***

ftn-6005 ftn: SCALAR File = resid.f, Line = 37

A loop starting at line 37 was unrolled 4 times.

ftn-6204 ftn: VECTOR File = resid.f, Line = 37

*A loop starting at **line 37 was vectorized.***

Cray Opteron Compiler: Current Weaknesses

- Tuned Performance
  - Vectorization
  - Non-temporal caching
  - Blocking
  - Many end-cases
- Scheduling
- Spilling
- No C++

- Very young X86 compiler

Cray Opteron Compiler: Future Capabilities

- C++ Support
- Optimized PGAS
  - Will require Gemini network to really go fast
- Improved Vectorization
- Automatic Parallelization
  - Modernized version of Cray X1 streaming capability
  - Interacts with OMP directives
- Improve Cache optimizations

Cray Opteron Compiler: Summary

- A new compiler which supplements available compilers
- Optimized CA-Fortran and UPC for future Cray products
  - Functional PGAS compiler avail on XT today
- Still a very young compiler
- With many interesting capabilities